

Appendix A

```

5 void MusicAspectIntegrator::Compute(void) {
    Value = 0;

    if (Values[STRUM_DENSITY] > 64) { /* strumming too fast */
        Value = -MAX_CHOOSER_VALUE; /* considered very bad */
10    } else if (Values[MELODY_PRESENT]) { /* must accompany singer */

        /* pick the most polyphonic (strummed chords, etc.) */
        Value += Values[POLYPHONY]*(MAX_CHOOSER_VALUE/10);

15    /* two strums per second. */
    #define FAVORITE_STRUM_DENSITY 16

        /* make value smaller if strum speed is off optimum */
        if (Values[STRUM_DENSITY] > FAVORITE_STRUM_DENSITY) {
20            Value = Value * FAVORITE_STRUM_DENSITY / Values[STRUM_DENSITY];
        } else {
            Value = Value * Values[STRUM_DENSITY] / FAVORITE_STRUM_DENSITY;
        }

25    } else { /* no singer. Give lead/melodic parts higher values */

        Value = 100;

        if (Values[STRUM_DENSITY] > FAVORITE_STRUM_DENSITY) {
30            Value = Value * FAVORITE_STRUM_DENSITY / Values[STRUM_DENSITY];
        } else {
            Value = Value * Values[STRUM_DENSITY] / FAVORITE_STRUM_DENSITY;
        }

35    if (Values[AVERAGE_PITCH] < 45) {
        Value = 0; /* pitch too low for lead. */
    } else {
        /* pick the most "interesting" */
40        Value += ABS(Values[PITCH_ACCELERATION])/8;

        /* emphasize loud parts */
        Value = Value * (100 + Values[LOUDNESS]) / 100;

45    }
    }
}

```

Appendix B

```

5  class MTRcEventNode *MTRcDerivative::GiveEventNode(void) {
    if (FirstTime) {
        FirstTime = FALSE;
        Node1 = SourceFilt->GiveEventNode();
10
        /* be tolerant of bad input.  Input must have at least one
           measurement.  But if not, we will still do what is right.*/
        /* we MUST return a measurement at time zero.  Make it zero.  */
        if (!Node1) return new MTRcEventNode(0, 0);
15    }

    if (!Node1) return NULL;    /* done  */

    /* if we have a Node1, we will return a measurement.  */
20    MTRcEventNode *node2 = SourceFilt->GiveEventNode();

    slong nodeval;

    if (node2) {    /* the measurement will be a function of both nodes  */
25        slong vdelta;

        vdelta = node2->GiveValue() - Node1->GiveValue();

        slong tdelta = node2->GiveTicks() - Node1->GiveTicks();
30        if (tdelta) {
            nodeval = vdelta * 16 * GlobalInputFile->GiveDivision() / tdelta;
        } else {
            nodeval = 0;
35        }
    } else {
        nodeval = 0;    /* zero, since we don't have two nodes.  */
    }

40    MTRcEventNode *retval = new MTRcEventNode(Node1->GiveTicks(), nodeval);

    delete Node1;
    Node1 = node2;

45    return retval;
}

```

20198155.doc